08 | 组件化：如何像搭积木一样开发网页？

2021-11-03 大圣

《玩转Vue 3全家桶》 课程介绍 >

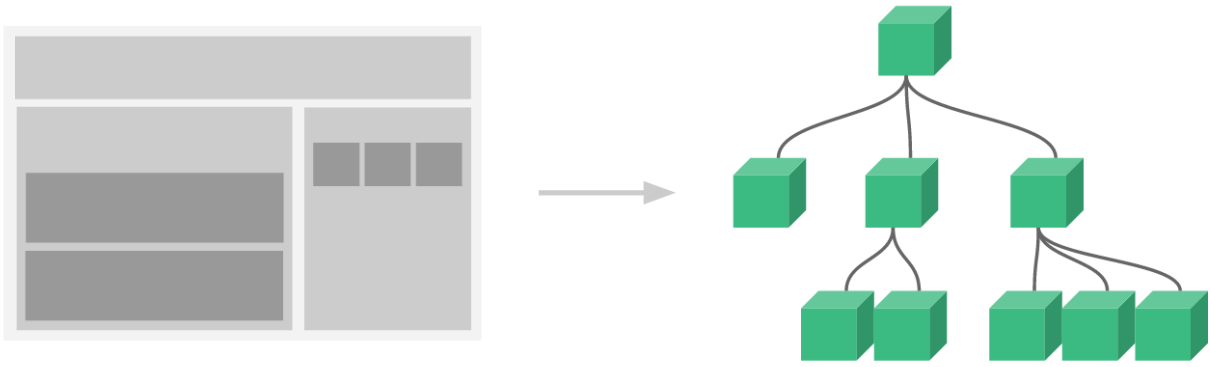
讲述：大圣
时长 10:45 大小 9.86M

你好，我是大圣。

在上一讲中，我给你讲解了响应式的基本原理和进阶用法。除了响应式，组件相关的知识在 Vue 中也非常重要，所以今天我就跟你聊一下 Vue 的组件化机制。

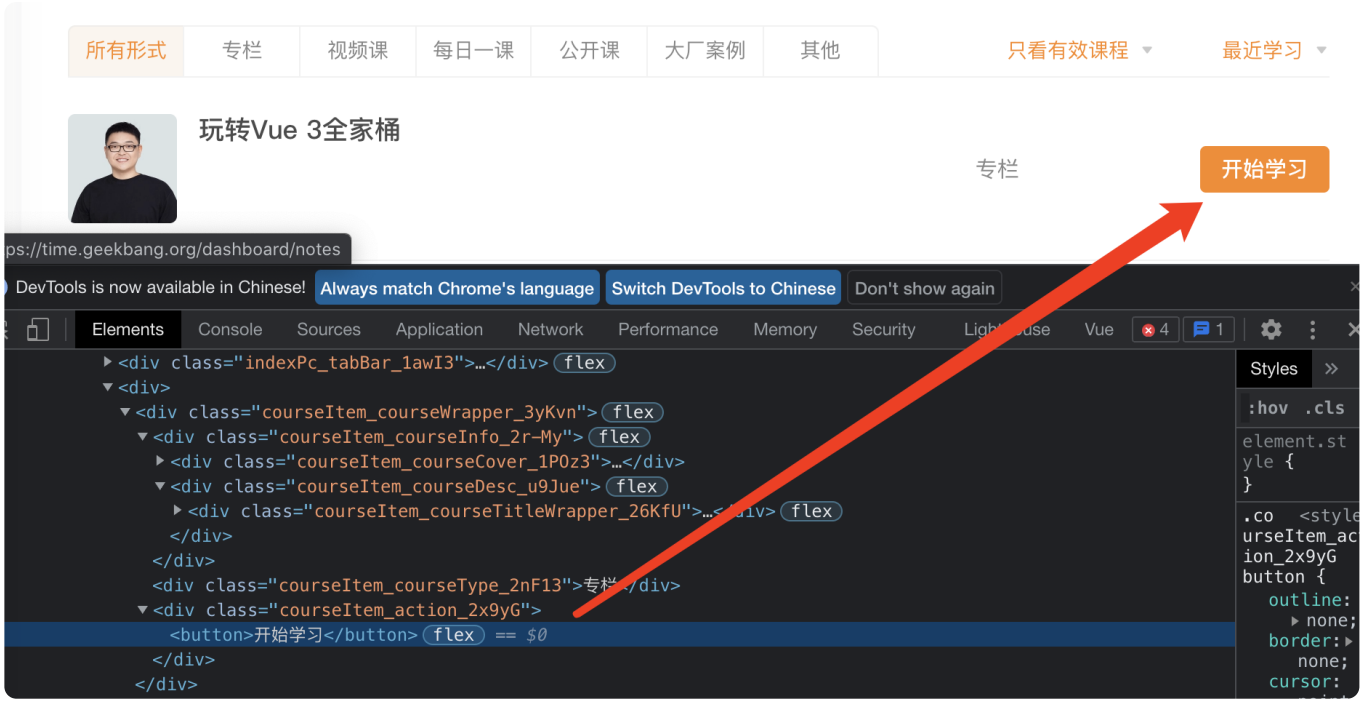
在我们的项目中，组件无处不在，通过对组件功能的封装，可以像搭积木一样开发网页。而我们现在已经很难想象，没有组件的开发状态是什么样了。你可以看下面 Vue 官方的例图，它对组件化开发做了形象化的展示。图中的左边是一个网页，可以按照功能模块把它想象成很多组件，这些组件就像积木一样拼接成网页。

领资料



什么是组件化开发

谈组件化开发之前，我们先来看看什么是组件。举个通俗的例子，我们在页面的源码里写出的 button 标签，会在前端页面中显示出下面的样式：

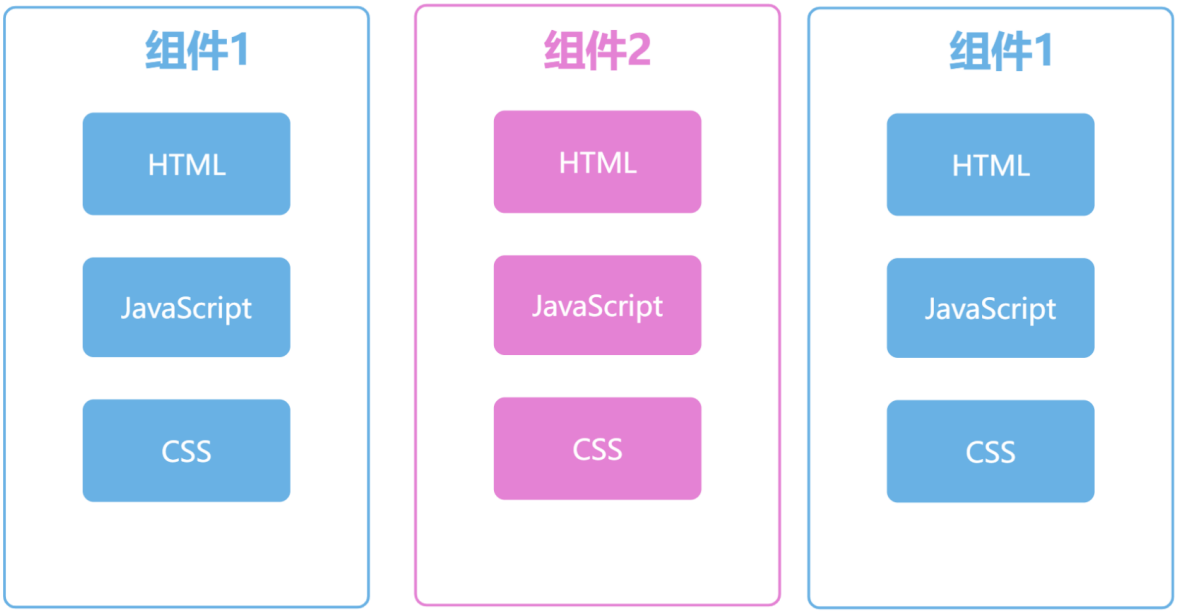


这个 button 其实就是一个组件，这样前端页面在显示上会加上边框和鼠标悬停的样式，还可以使用 click 事件触发函数等。只不过这是浏览器帮我们封装好的组件，我们在编辑代码的任何地方，只需要使用下面的代码，就可以让前端页面显示一个按钮。

复制代码

```
<button> 按钮 </button>
```

除了浏览器自带的组件外，Vue 还允许我们自定义组件，把一个功能的模板（template）封装在一个.vue 文件中。例如在下图中，我们把每个组件的逻辑和样式，也就是把 JavaScript 和 CSS 封装在一起，方便在项目中复用整个组件的代码。



极客时间

我们实际开发的项目中有导航、侧边栏、表格、弹窗等组件，并且也会引入 Element3 这个组件库进行开发。此外，我们也会定制业务相关的组件，最终通过这些组件，搭积木式地把页面搭建起来。

Vue 已经把组件化的机制实现得很好了，你只需要在这个基础之上，去掌握和学习组件化在使用上的设计理念。这样做的目的是实现高效的代码复用，在后续的项目开发中，我们会把组件分成两个类型，一个是通用型组件，一个是业务型组件。

通用型组件就是各大组件库的组件风格，包括按钮、表单、弹窗等通用功能。业务型组件包含业务的交互逻辑，包括购物车、登录注册等，会和我们不同的业务强绑定。

组件的开发由于要考虑代码的复用性，会比通常的业务开发要求更高，需要有更好的可维护性和稳定性的要求。为了帮助你理解设计组件的要点，我先选择一个简单的组件展开讲解。小圣在继续开发项目的时候，有一个评级的需求，简单来说，就是在前端页面上，能够让商品显示 1 到 5 的评分。我会借此教小圣在组件开发上的入门和应用，并依次讲解组件的设计思路。

渲染评级分数

其实，对于简单的评级需求，我们就可以使用组件。这样，只需要一行代码就可以实现评级需求。比如下面的代码，rate 是 1 到 5 的整数，通过 slice 方法，我们直接渲染出对应数量的星星即可。

```
"★★★★★☆☆☆☆☆".slice(5 - rate, 10 - rate)
```

复制代码

想要查看上面这行代码的效果，你只需要传入评分值 rate。这行代码的运行效果如下图所示，其中的星星代表着评价的等级，由 rate 的值来决定。

每个组件渲染的内容并不完全一样，这是我们写组件时首先要确认的内容。每个组件在项目中的不同地方，会渲染不同的内容。

我们在这里写的这个组件就是根据 rate 的值，来渲染出不同数量的星星。我们进入到 src/components 目录，新建 Rate.vue，然后写出下面的代码。在下面的代码中，我们使用 defineProps 来规范传递数据的格式，这里规定了组件会接收外部传来的 value 属性，并且只能是数字，然后根据 value 的值计算出评分的星星。

复制代码

```
<template>
    <div>
        {{rate}}
    </div>
</template>

<script setup>
import { defineProps,computed } from 'vue';
let props = defineProps({
    value: Number
})
let rate = computed(()=>"★★★★★☆☆☆☆☆".slice(5 - props.value, 10 - props.val
</script>
```

使用组件的方式就是使用:value 的方式，通过属性把 score 传递给 Rate 组件，就能够在页面上根据 score 的值，显示出三颗实心的星星。下面的代码展示了如何使用 Rate 组件来显示 3 颗星星。

复制代码

```
<template>
<Rate :value="score"></Rate>
</template>

<script setup>
import {ref} from 'vue'
import Rate from './components/Rate1.vue'
let score = ref(3)
</script>
```

根据传递的 score 值显示的不同的内容，我们也可以更进一步，回到 Rate.vue 代码里，加入如下的代码，比如在组件中内置一些主题颜色，加入 CSS 的内容。如下面代码，Rate 组件新接收一个属性 theme，默认值是 orange。我们在 Rate 组件中内置了几个主题颜色，根据传递的 theme 计算出颜色，并且使用 :style 渲染。

复制代码

```
<template>
    <div :style="fontstyle">
        {{rate}}
    </div>
</template>

<script setup>
import { defineProps,computed, } from 'vue';
let props = defineProps({
    value: Number,
    theme:{type:String,default:'orange'}
})
console.log(props)
let rate = computed(()=>"★★★★★☆☆☆☆☆".slice(5 - props.value, 10 - props.val
const themeObj = {
  'black': '#00',
  'white': '#fff',
  'red': '#f5222d',
  'orange': '#fa541c',
  'yellow': '#fadb14',
  'green': '#73d13d',
  'blue': '#40a9ff',
}
const fontstyle = computed(()=> {
    return `color:${themeObj[props.theme]};`
})

</script>
```

在完成了上面代码所示的这一过程，也就是通过 theme 渲染星星颜色这一步，我们就可以使用下面的代码，传递 value 和 theme 两个属性，并且可以很方便地复用组件。

复制代码

```
<Rate :value="3" ></Rate>
<Rate :value="4" theme="red"></Rate>
<Rate :value="1" theme="green"></Rate>
```

在下图中，也可以看到上面三个组件渲染的结果：

组件事件

使用这个 Rate 组件后，虽然前端页面显示评级的需求是完成了，但是评级组件还需要具备修改评分的功能，所以我们需要让组件的星星可点击，并且让点击后的评分值能够传递到父组件。

在 Vue 中，我们使用 emit 来对外传递事件，这样父元素就可以监听 Rate 组件内部的变化。现在我们对 Rate 组件进行改造，首先由于我们的星星都是普通的文本，没有办法单独绑定 click 事件。所以我们要对模板进行改造，每个星星都用 span 包裹，并且我们可以用 width 属性控制宽度，支持小数的评分显示。

我们回到 Rate.vue 组件，添加下面的代码，我们把★和☆用 span 包裹，并绑定鼠标的 mouseover 事件。然后通过:style，我们可以设置实心五角星★的宽度，实现一样的评级效果。

复制代码

```
<template>
<div :style="fontstyle">
    <div class='rate' @mouseout="mouseOut">
      <span @mouseover="mouseOver(num)"  v-for='num in 5' :key="num">☆</span>
      <span class='hollow' :style="fontwidth">
        <span @mouseover="mouseOver(num)" v-for='num in 5' :key="num">★</span>
      </span>
    </div>
</div>
</template>
<script setup>
// ...其他代码
// 评分宽度
let width = ref(props.value)
function mouseOver(i){
    width.value = i
}
function mouseOut(){
```

```
    width.value = props.value
}
const fontwidth = computed(()=>`width:${width.value}em;`)

</script>

<style scoped>
.rate{
  position:relative;
  display: inline-block;
}
.rate > span.hollow {
  position:absolute;
  display: inline-block;
  top:0;
  left:0;
  width:0;
  overflow:hidden;
}
</style>
```

因为现在是通过宽度显示星星，所以我们还可以支持 3.5 分的小数评级，并且支持鼠标滑过的时候选择不同的评分。用下面的代码，我们可以使用 Rate 组件。

复制代码

```
<Rate :value="3.5" ></Rate>
```

上面代码的显示效果如下图所示：

然后我们需要做的，就是在点击五角星选择评分的时候，把当前评分传递给父组件即可。在 Vue 3 中，我们使用 defineEmit 来定义对外“发射”的数据，在点击评分的时候触发即可。下面的 defineEmit 代码就展示了点击评分后，向父元素“发射”评分数据 num。

复制代码

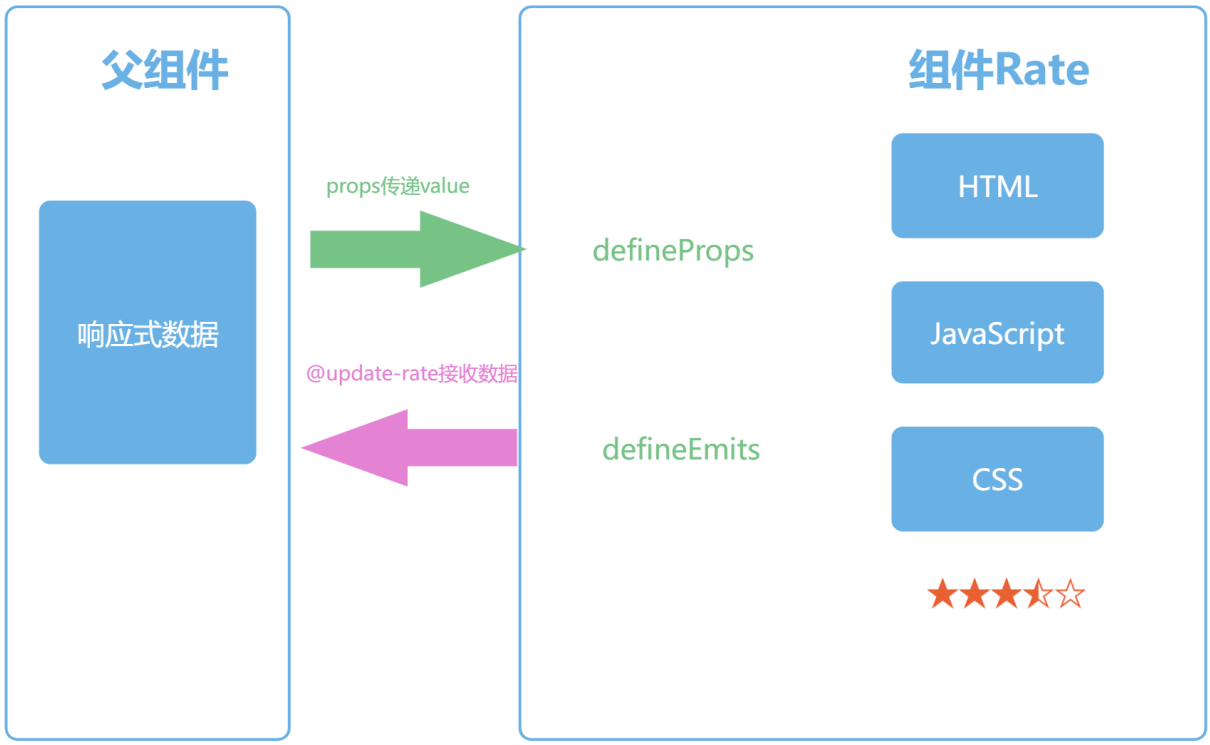
```
<template>
  省略代码
   <span @click="onRate(num)" @mouseover="mouseOver(num)" v-for='num in 5' :ke

</template>
<script setup>
import { defineProps, defineEmits,computed, ref} from 'vue';

let emits = defineEmits('update-rate') // 定义emits
function onRate(num){
    emits('update-rate',num)
}
</script>
```

在下面的代码中，我们使用 @update-rate 接收 Rate 组件 emit 的数据，并且修改 score 的值，这样就完成了数据修改后的更新。

复制代码

```
<template>

<h1>你的评分是 {{score}}</h1>
<Rate :value="score" @update-rate="update"></Rate>

</template>

<script setup>
import {ref} from 'vue'
import Rate from './components/Rate1.vue'
let score = ref(3.5)
function update(num){
    score.value = num
}

</script>
```

现在组件的示意图如下，我们通过 defineProps 定义了传递数据的格式，通过 defineEmits 定义了监听的函数，最终实现了组件和外部数据之间的同步。



组件的 v-model

上面 Rate 组件中数据双向同步的需求在表单领域非常常见，例如第二讲中，我们在 input 标签上使用 v-model 这个属性就实现了这个需求。在自定义组件上我们也可以用 v-model，对于自定义组件来说，v-model 是两个语法，也就是：传入属性和接收组件传递数据和的简写。

在下面的代码中，首先我们把属性名修改成 modelValue，然后如果我们想在前端页面进行点击评级的操作，我们只需要通过 update:modelValue 这个 emit 事件发出通知即可。

复制代码

```
let props = defineProps({
    modelValue: Number,
    theme:{type:String,default:'orange'}
})
let emits = defineEmits(['update:modelValue'])
```

然后我们就可以按如下代码中的方式，使用 Rate 这个组件，也就是直接使用 v-model 绑定 score 变量。这样，就可以实现 value 和 onRate 两个属性的效果。

复制代码

```
<template>

<h1>你的评分是 {{score}}</h1>
<Rate v-model="score"></Rate>
</template>
```

插槽

和 HTML 的标签使用类似，很多时候我们也需要给组件中传递内容。就像在下面的代码中 click 并不是 button 标签的属性，而是子元素，button 标签会把子元素渲染在居中的位置。

复制代码

```
<button> click </button>
```

我们的 Rate 组件也是类似的，在 Vue 中直接使用 slot 组件来显示组件的子元素，也就是所谓的插槽。在下面的代码中，我们使用 slot 组件渲染 Rate 组件的子元素。

复制代码

```
<template>
<div :style="fontstyle">
    <slot></slot>
    <div class='rate' @mouseout="mouseOut">
      <span @mouseover="mouseOver(num)"  v-for='num in 5' :key="num">☆</span>
      <span class='hollow' :style="fontwidth">
        <span @click="onRate(num)" @mouseover="mouseOver(num)" v-for='num in 5
      </span>
    </div>
</div>
</template>
```

我们现在使用 Rate 组件的时候，组件的子元素都会放在评级组件之前。除了文本，也可以传递其他组件或者 html 标签 。下面代码显示的结果，第一个 Rate 组件显示课程评分 ，第二个 Rate 组件前面显示一个图片。

复制代码

```
<Rate v-model="score">课程评分</Rate>
```

```
<Rate v-model="score">
    <img width="14" src="/favicon.ico">
</Rate>
```

总结

至此，我们设计了一个非常迷你的评级组件，我带你回顾一下今天都做了什么吧。

首先，我们用 props 属性传递的方式，通过传递的 value 属性去决定显示的星星数量，这也是设计组件的时候首先就要考虑的。一个组件库首先要实现的，就是通过 props 属性渲染内容，而在 Rate 组件里，我们可以根据 value 属性去渲染显示了几颗星星。

然后为了让用户可以点击评分，我们优化了显示的方式，每个★包裹一个 span 标签，并且绑定了 mouseover 和 mouseout 事件显示鼠标悬停的样式，最后在★标签的 click 事件上，对外通知事件，告知父组件数据的变化。

对于这个通知机制，我们使用 defineEmits 定义的方式来实现，这也是 Vue 组件中重要的数据交换机制。emits 配合 props，这样我们在使用一个组件的时候，就实现了给组件传递数据，并且我们也能够监听组件内部数据的变化。最后我们通过规范 props 和 emit 的名字，实现了直接在自定义的组件之上使用 v-model。

当然，这个组件开发到这里，依然是比较精简，但实际上我们要考虑的问题更多。比如在下一讲，我会给你讲解如何在 Vue 中加上一些简单的动画。字符★显示的效果可能也不符合你所在公司的设计规范，所以你可能就需要使用图片替换五角星样式等。

更加完备的 rate 组件，你可以去看 Element3 评级组件的实现，在这里我们先初步感受一个组件的完整设计过程，它包含着交互的小细节，比如键盘事件、自定义 icon 等等。

Rate 组件写完，Vue 的组件化你就算入门了，在课程第三部分中，我们会对更多场景的组件化进行实战开发，在那里我会给你详细说说更多组件进阶和扩展的用法。

思考题

今天我带你从 0 设计了一个组件，相信你对 Vue 3 的组件设计也有了新的思考，那么关于这个 Rate 组件，你觉得还有哪些功能扩展的需求呢？

欢迎在评论区分享你的思考，也欢迎你把这一讲分享给其他人，我们下一讲见！

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 20 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 07 | 巧妙的响应式：深入理解Vue 3的响应式机制

下一篇 09 | 动画：Vue 中如何实现动画效果？

精选留言 (30)

写留言

II

2021-11-03

1. 对组件化的理解

组件化好似“搭积木”；“分而治之”思想实际运用，所谓“大事化小，小事化了”或称为“抽

象隔离”。就是，各个组件之间有自己需要解决的问题，有各自解决的方式方法，但互相

不需要知道，它们沟通只看“结果”即 props，events（当然这是模拟事件，也是个可以...

展开

作者回复: cool 好顶赞

1 9

bugu

2021-11-03

v-model那一部分，根据文中代码没实现出来

展开

作者回复: function onRate(num){

emits('update:modelValue',num)

}

这一段吗 sorry

1 5

cskang

2021-11-03

代码能放到 CodePen 上就好了，这样方便查看完整的代码，又可以通过修改调试学习。

4

peterpc

2021-11-04

点击click事件@click="onRate(num)"，浏览器调试提示如下警告，是什么原因：

[Vue warn]: Component emitted event "update:modelValue" but it is neither declared in the emits option nor as an "onUpdate:modelValue" prop.

另外在chrome和firefox都出现五角星显示错位的情况

展开

2 3

费城的二鹏

2021-11-03

第一个版本的评分组件，简直是简洁的巅峰，自己没想到可以这么做，看到这个逻辑秒

懂，太精巧了。

作者回复: 但是用这个符号不够精致，用来演示还是够了

3

请去学习吧
2021-11-05

被"★★★★★☆☆☆☆☆".slice(5 - props.value, 10 - props.value)惊艳到了，在这个课程里收获了好多惊喜

作者回复: 感谢感谢

2

费城的二鹏
2021-11-03

1. 代码目前都是一个颜色，能否添加关键词变色，方便我们阅读？
2. 对代码的增量修改能否通过颜色或者其它方式的变动来指示哪些是新增的，哪些是旧的，方便读者阅读出新增部分代码。

展开

作者回复: 代码高亮吗？这个得给极客时间提个需求去

2

The Word
2021-11-05

3.2以后defineEmits和difineProps都宏编译了，不需要引入了，defineEmit已经不能用了

作者回复: 3.2的更新后面我会补充一下，并且3.2针对ref的value属性也做了优化

1

Geek_1abc92
2021-11-04

每一章能否贴一下完整的代码？

展开

1

Geek_40b392
2021-11-04

为什么创建是的Rate.vue,但是引入的时候是Rate1.vue。使用组件的那个文件也没有讲在哪里创建。

作者回复: Rate1是个手误。。囧。默认这段教程的使用都是在入口组件演示的

1

pawel
2021-11-03

let emits = defineEmits('update-rate') // 定义emits
大圣老师，defineEmits好像只能接受Array<string> | Object

展开

1

Mserke
2021-11-06

如果子组件想要改变父组件传递参数的值，请问该怎么处理呢？
只能用ref 新建一个变量吗，有更好的处理方式吗？

展开

Mserke
2021-11-06

如果子组件想要改变父组件传递参数的值，应该怎么处理呢？

Alias
2021-11-05

"vue": "^3.2.21"
1 v-model="score" 写法，子组件接收时必须用 modelValue
2 除非是 v-model:xxx="score" 才能自定义
3 子组件接收后 props 不能解构。。否则mouseout 后又变成初始传进来的值。。。
const {value, theme} = props...

展开

Geek_7cdaba
2021-11-05

建议作者把代码弄全一点，v-model那敲了一遍还是没弄明白，呜呜呜

NiceBlueChai
2021-11-05

Component emitted event "update-rate" but it is neither declared in the emits option nor as an "onUpdate-rate" prop.

展开

作者回复: 代码方便发出来看下吗

1

轻飘飘过
2021-11-05

mac safari上星星显示正常，chrome上显示少了，了解到是em在不同浏览器中的问题，这个单位怎么兼容呢😀

展开

作者回复: 写死字体大小 过几天我更新一下代码把

peterpc
2021-11-05

v-model双向绑定笔记：
如果子组件属性名定义为modelValue，则父组件直接v-model="父组件属性"即可绑定，如果子组件需要自定义属性名称，比如定义为abc，则父组件绑定时必须按照这种写法 v-model:abc="父组件属性"才能绑定。
如下实例：...

展开

1

王志成
2021-11-05

可以把课程中的代码开源出来吗？vue小白，看着教程写代码，有些地方运行不下去。

gongshun

2021-11-05

vue3官网的 `v-model` 的语法是这样的，你那个是 vue2 的写法吧

```
<my-component v-model:title="bookTitle"></my-component>
```

展开

1